Modern Cryptanalysis of Schlüsselgerät 41

George Lasry
The CrypTool Team
george.lasry@cryptool.org

Abstract

The Schlüsselgerät 41 was an highly secure encryption machine developed by Fritz Menzer and used from 1944 by the Abwehr. Bletchley Park could not decipher its traffic. In this article, we provide a functional description of the SG-41 and present a novel cryptanalytic method to recover the key settings from ciphertext and known-plaintext. This attack requires extensive computing power, a testimony to the resilience of the SG-41 even against modern cryptanalysis. We also present an alternative method, based on acoustic cryptanalysis, which allows for the recovery of the key settings in minutes.

1 Overview

This article is structured as follows: In Section 2, a brief overview of the history of the SG-41 is given and a functional description. Section 3 describes cryptanalysis attempts by Bletchley Park and the US against the SG-41. In Section 4, we describe a novel known-plaintext attack that is feasible but requires extensive computing power, and in Section 5, a highly-efficient side-channel attack that relies on acoustic analysis of the device. Finally, in Section 6, we assess the security of the SG-41 compared to other encryption machines of the 1940s.

2 The SG-41 – Introduction

The SG-41 was an encryption machine introduced by Fritz Menzer, Regierungsoberinspektor of OKW/Chi, the cryptographic branch of the Wehrmacht. While inspired by the Hagelin pin-and-lug devices, the design of the SG-41 incorporated several novel features that significantly enhanced its security. Logistical reasons prevented its production in large volumes, and it was only deployed in 1944 on a few Abwehr networks. Bletchley Park could not decipher its traffic unless multiple messages were sent in-depth. Until recently, little was known about the inner functioning of the SG-41. Several historical documents have been declassified that provide extensive details about its functioning. A small number of SG-41 have survived, and some have been restored.

In this section, we provide an overview of the history of the SG-41, as well as a functional description, and an analysis of its keyspace size.

2.1 Fritz Menzer and the SG-41

Fritz Menzer (1908–2005) was the Government Inspector (Regierungsoberinspektor) of OKW/Chi, the cryptographic arm of the Wehrmacht, and later, Admiral Canaris, the head of the Abwehr charged him with ensuring the security of the organization's communications. Menzer designed and led the development of several cipher devices, methods, and procedures, some of which created some difficulties for British and U.S. codebreakers. In a post-war NSA publication, Menzer is described as "Cryptologic Inventor Extraordinaire", and the peak of his achievements, however, is most probably the invention of the SG-41, shown in Figure 1 (Mowry, 1983).

Having previously worked on the cryptanalysis of the Enigma and the Hagelin C-36, Menzer understood their weaknesses. While much of the SG-41 borrows from the Hagelin pin-and-lug design, Menzer introduced some features that provided enhanced security (Mowry, 1983). Boris Hagelin later complained to William Friedman that the Germans had stolen his design. He had obtained one of the SG-41 machines, wrongly calling it C-41 (Friedman, 1955).[1]

The SG-41 was designed with a keyboard and a strip printer to speed up the process of enciphering

[1] It is also called C-41 in some U.S. documents (Agency, 1947).

and deciphering (unlike the Enigma that required at least two operators, one to type into the keyboard, and another one to write down the lamps activated).

The army ordered 11,000 units in 1942, and a prototype was presented in 1943, but by the end of the war, only 1000–1500 had been produced by the firm Wanderer-Werke in Chemnitz. The challenges of wartime production and the lack of material may have prevented its production in higher volumes. In addition, the device was considered too heavy - over 13 kg - to be used at the frontlines (Dahlke, 2018; Mowry, 1983). Near the end of 1944, it was deployed on at least three Abwehr links, between Berlin, Bordeaux in Southern France, Northern Italy, and Vienna, replacing the Enigma G machines (Batey et al., 1945).

Figure 1: The SG-41

2.2 Functional Description of the SG-41

Until recently, little was known about the internal mechanism of the SG-41 (Dahlke, 2018; Museum, 2020b; Schmeh, 2004). Recently declassified U.S. and British documents, and in particular, a wartime report from Bletchley Park, provide enough details to fully reconstruct the functioning of the SG-41 (Batey et al., 1945; Mowry, 1983; Mowry, 1989; Mowry, 2003). While very few devices have survived, most in bad condition, some units are in the hands of museum curators and crypto collectors, who were able to analyze the physical/mechanical design of the SG-41. At least one machine has been restored so that it is fully functional (Historica, 2019; Dahlke, 2018).

While the SG-41 is described in several documents (Mowry, 1989; Mowry, 2003; Mowry, 1983; WDGAS-14, 1946), those descriptions are incomplete, and sometimes conflicting. The most reliable historical source describing the SG-41 is a G.C. & C.S. report titled *Secret Service SIGINT Volume II - Cryptographic Systems and Their Solutions - Machine Cyphers* written by Keith Batey, Mavis Batey, Margaret Rock, and Peter Twinn in 1945. The authors were part of ISK - Intelligence Services Knox (headed by Dilly Knox before his death in 1943) and were responsible for analyzing Abwehr traffic with its agents and offices worldwide. While most of the report is about the cryptanalysis of the Abwehr Enigmas, against which ISK had considerable success, the last seven pages of the report are dedicated to a detailed functional description of the SG-41 and to the mostly unsuccessful attempts by ISK to decipher its traffic (Batey et al., 1945).

The focus in this section is on the logical and functional aspects of the SG-41, rather than on its physical design and implementation. Figure 2 shows a functional diagram of the SG-41. The SG-41 enciphers symbols of the A-Z alphabet into symbols of the same alphabet. To encipher, the operator presses a plaintext symbol on the keyboard (spaces are represented by the symbol J). The plaintext symbol is encrypted, and the resulting ciphertext symbol is printed on a paper strip (together with the plaintext symbol). The decryption process is similar: The operator presses a ciphertext symbol on the keyboard. The encryption process, which is reciprocal, converts back the ciphertext symbol into a plaintext symbol printed on the paper strip (together with the ciphertext symbol).

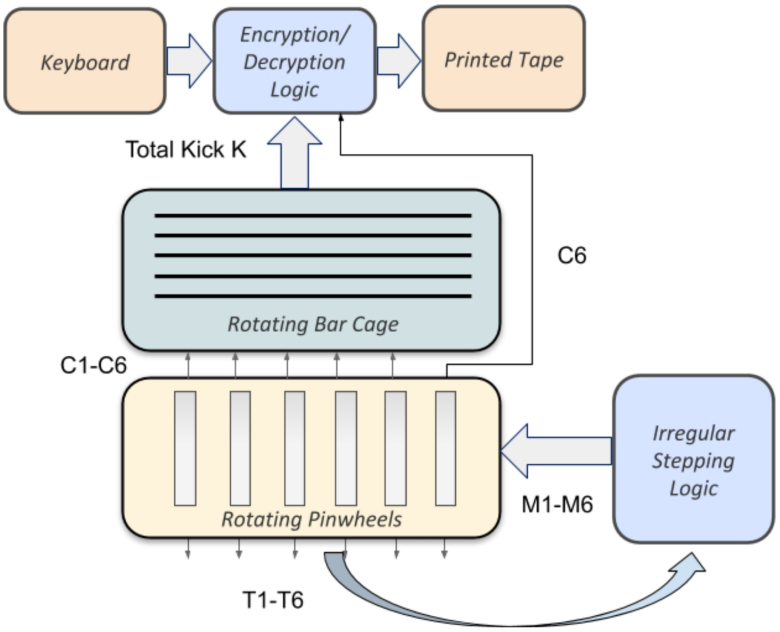

Figure 2: The SG-41 - Functional Diagram

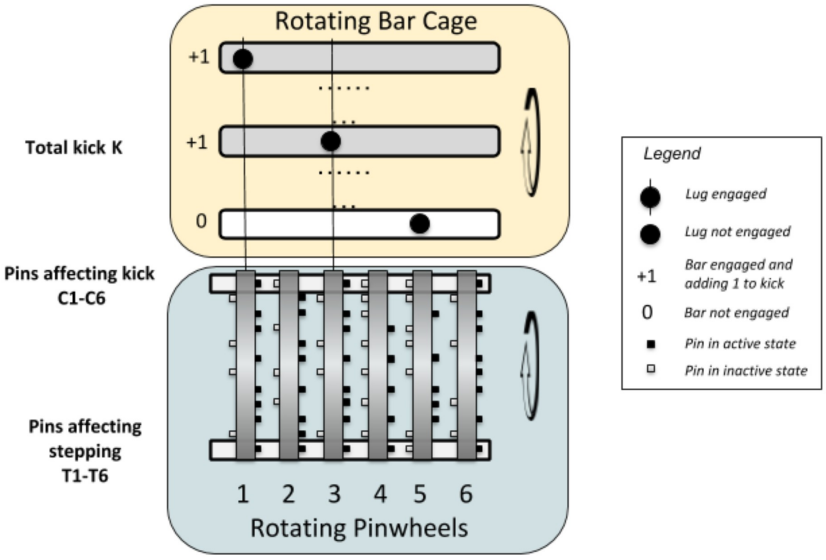

Figure 3: The SG-41 - Functional Diagram - Pinwheels and Bar Cage

The encryption logic is governed by the outputs of a *rotating bar cage* with 25 bars, as well as of a set of six rotating *pinwheels*, as shown in Figure 3. The pinwheels are numbered 1 to 6, from left to right, and have 25, 25, 23, 23, 24, and 24 pins each, respectively. Each pin can be set to an *active* or an *inactive* state. Each wheel from 1 to 5 affects one or more bars. The pin currently in front of the cage determines whether the bar is engaged or not. Those pins are denoted as C1 to C5, for wheels 1 to 5, respectively. C6 does not affect the cage bars, but it affects encryption as described in Section 2.4.

Each bar has a fixed lug positioned in front of one of the wheels 1 to 5. In front of wheels 1, 2, 3, 4, and 5, there are 1, 2, 4, 8, and 10 bars with such a fixed lug, respectively. The bar cage performs a full rotation during the encryption of a single symbol. When a bar has its lug against an active pin, it is engaged and it adds one to a total additive kick, denoted as K (its function is described in Section 2.4). If a bar is not engaged, it does not add to K. Therefore, wheels 1 to 5 may add 1, 2, 4, 8, and 10, to K, respectively. When all the bars are against active pins, and thus, they all are engaged, the total kick is $1+2+4+8+10=25$.

So far, the mechanism of the first five wheels and the bar cage is very similar to the Hagelin C-35, which also had bars with fixed lugs, and five wheels that affect kick (Museum, 2020a). But the SG-41 introduces two features which greatly enhance its cryptographic security: irregular stepping (see Section 2.3), and complementary kick (see Section 2.4). It also features a fixed substitution (see Section 2.5), which has no significant effect on cryptographic security.

2.3 Irregular Stepping

The SG-41 features an irregular wheel stepping mechanism. The stepping of each wheel is governed by the state of pins in other wheels (see Figure 2). The pins of wheels 1 to 6 that affect other wheels' stepping are denoted as T1 to T6. Those pins are at a certain distance on the wheel from the pins that control the bars (C1-C6). With wheel 1, if pin 1 currently affects the bar cage (it generates C1), at the same time, pin 14 affects the stepping of another wheel (generating T1). Similarly, if the wheel has advanced one step, pin 2 generates C1 and pin 15 generates T1, and so forth. The same applies to C2 and T2 for wheel 2, and similarly to the other wheels.

A full *encryption cycle*, in which a plaintext symbol is encrypted (or a ciphertext symbol is decrypted), and the wheels advance, consists of three stages, one of which is optional (Batey et al., 1945):

1. An optional *pre-encryption stepping stage* that occurs before encryption, only if T6 was active (at the beginning of the cycle).
2. The *encryption stage*.
3. A *post-encryption stepping stage* that always occurs after encryption.

The two stepping stages are identical. Each stepping stage consists of two phases, as follows:

- Wheels 2 to 6 step if the pin affecting stepping on the wheel on its left was active (at the beginning of the stage). For example, if T3 was active, wheel 4 steps.
- All the wheels (1 to 6) step.

This mechanism creates a circular interdependence between the wheels, as illustrated in Figure 4. This circular interdependence means that any wheel may affect the stepping of any other wheel, directly or indirectly, and that there is no way to know how the wheels step without first determining the pin settings of all wheels.

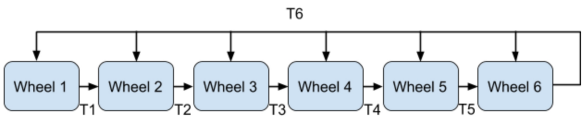

Figure 4: Wheel Stepping – Circular Interdependency

This two-stage mechanism ensures that every wheel will step at least once, and no more than four times, per encryption cycle. Depending on T6 at the beginning of the encryption cycle, either:

- Wheels 2 to 6 step two to four times, and wheel 1 steps twice, or
- Wheels 2 to 6 step once or twice, and wheel 1 steps once.

Wheel 2 has 25 pins and completes a full rotation after 7 to 13 cycles. Wheels 3, 4, 5, and 6 (with 24 or 23 pins) complete a full rotation after 6 to 12 cycles.

2.4 Complementary Kick

The second security enhancement has to do with how K affects encryption. In the regular Hagelin C machines, encryption is according to the Beaufort reciprocal formula (p is the plaintext symbol, c the resulting ciphertext symbol, and K the total kick) (Lasry et al., 2016):

$$c = (25 - p + K) \mod 26 \quad (1)$$

Decryption works similarly (modulo 26):

$$p = 25 - c + K = 25 - (25 - p + K) + K = p \quad (2)$$

The SG-41 introduces a *complementary feature*, that works as follows (modulo 26):

- If C6 is inactive: $c = 25 - p + K$
- If C6 is active: $c = 25 - p + (25 - K)$, effectively complementing the kick.

The complementary feature complicates the relationship between the effective kick K_e, computed as $K_e = c + p - 25$, and the state of C1-C5, the pins that affect encryption. Without the complementary feature, if $K_e = 1$, for example, we could clearly establish that C1 is active, and C2-C6 are inactive. But with the complementary feature, $K_e = 1$ could also be obtained if C1 is inactive, C6 is active, and C2-C5 are active. In general, there are two possible C1-C6 options for any K_e between 0 and 9, and from 16 to 25 (instead of one without the complementary feature). Similarly, there are four possible C1-C6 options for any K_e between 10 and 15 (instead of two without the complementary feature), as illustrated in Table 1.

2.5 Fixed Substitution

It should be noted that SG-41 first applies a substitution alphabet (denoted as S) to the input symbol, and its inverse to the output symbol, after encryption. The substitution alphabet is as follows (the letter on the top row maps to the letter on the bottom row, e.g., A maps to P, B maps to T, etc...):

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
PTOIUHVRFWACXQSEZKGMYJBNLD
```

So therefore the full encryption formula (modulo 26) is as follows:

- If C6 is inactive: $c = S^{-1}(25 - S(p) + K)$
- If C6 is active: $c = S^{-1}(25 - S(p) + (25 - K))$

Effective Kick K_e	**Active C1-C6**
0	none active C1+C2+C3+C4+C5+C6
1	C1 C2+C3+C4+C5+C6
2	C2 C1+C3+C4+C5+C6
3	C1+C2 C3+C4+C5+C6
4	C3 C1+C2+C4+C5+C6
5	C1+C3 C2+C4+C5+C6
6	C2+C3 C1+C4+C5+C6
7	C1+C2+C3 C4+C5+C6
8	C4 C1+C2+C3+C5+C6
9	C1+C4 C2+C3+C5+C6
10	C5 C2+C4 C1+C3+C5+C6 C1+C2+C3+C4+C6
11	C1+C5 C1+C2+C4 C3+C5+C6 C2+C3+C4+C6
12	C3+C4 C2+C5 C1+C3+C4+C6 C1+C2+C5+C6
13	C1+C3+C4 C1+C2+C5 C3+C4+C6 C2+C5+C6
14	C3+C5 C2+C3+C4 C1+C5+C6 C1+C2+C4+C6
15	C1+C3+C5 C1+C2+C3+C4 C5+C6 C2+C4+C6
16	C2+C3+C5 C1+C4+C6
17	C1+C2+C3+C5 C4+C6
18	C4+C5 C1+C2+C3+C6
19	C1+C4+C5 C2+C3+C6
20	C2+C4+C5 C1+C3+C6
21	C1+C2+C4+C5 C3+C6
22	C3+C4+C5 C1+C2+C6
23	C1+C3+C4+C5 C2+C6
24	C2+C3+C4+C5 C1+C6
25	C1+C2+C3+C4+C5 C6

Table 1: Options for Effective Kick K_e

In a still-classified TICOM report, Menzer claims that this alphabet was designed to flatten the frequency counts in the ciphertext (Mowry, 1983; I-72, 1945). While it is true that the effective kick stream is not randomly distributed (the values 10 to 15 are more likely to appear), it is not clear to what extent this additional substitution enhances the cryptographic security of the SG-41.

2.6 Analysis of the Keyspace

Any pin on a pinwheel may be set to be either active or inactive. There are $25+25+23+23+24+24=144$ pins, therefore the size of the theoretical keyspace is 2^{144}. In practice, operational procedures on how to set the pins would probably have reduced this number. Unfortunately, no documents have survived that describe the operational procedures of the SG-41.

3 Historical Cryptanalysis of the SG-41

In this section, we present historical attempts at the Cryptanalysis of the SG-41.

3.1 Attacks on Depths

(Batey et al., 1945) describes how the mechanism of the SG-41 was reconstructed from depths by Bletchley Park, but could only be fully understood after a unit was captured in 1945. The SG-41, similarly to other Hagelin cipher machines, is still susceptible to attacks on messages in-depth, that is, encrypted with the same key settings. If we have two ciphertexts originally enciphered with the same key settings, and we look at the ciphertext symbols c_1 and c_2 at the same position in the message, then, assuming that C6 is inactive at that encryption cycle, we obtain (modulo 26):

$$c_1 = S^{-1}(25 - S(p_1) + K) \tag{3}$$

$$c_2 = S^{-1}(25 - S(p_2) + K) \tag{4}$$

where p_1 and p_2 are the corresponding unknown plaintext symbols.

After applying S (the known fixed substitution) on both sides, we obtain:

$$S(c_1) = 25 - S(p_1) + K \tag{5}$$

$$S(c_2) = 25 - S(p_2) + K \tag{6}$$

and therefore:

$$S(p_1) + S(c_1) = S(p_2) + S(c_2) \tag{7}$$

It can easily be seen that Equation 7 also applies if C6 is active and $c = S^{-1}(25 - S(p) + (25 - K))$.

Since S, c_1, and c_2 are known, if we can guess p_1, we obtain p_2.

The same techniques historically used for recovering depths (e.g., from Hagelin ciphertexts) can be applied here (Lasry et al., 2018).

Depths are available if operational discipline is poor, and the same key and starting positions are reused for different messages. From the historical reports, it can be understood that the same key settings (the active and inactive pins on the wheels) were used for a certain period of time. To avoid sending messages in-depth, the operator would first change the starting positions of the wheels for each message and securely communicate to the other party those starting positions, using concealed indicators (Batey et al., 1945).

3.2 Conditions for a Long Period

Another way depths may occur is if the machine repeats the keystream (the series of K_e) after a relatively short period, which we denote as *motion period*. In theory, because of irregular stepping, this should happen only after 25 * 25 * 23 * 23 * 24 * 24 = 190,440,000 stepping stages. In practice, the longest achievable period will be shorter than that, as there might be one or two stepping stages per encryption cycle.

An historical report by the predecessor to the NSA, the Army Security Agency, analyses the preconditions for a full motion period. According to the report, the Germans came up with a list of necessary and sufficient conditions to ensure a maximum period. The keys were selected to always comply with those conditions (Agency, 1947; I-72, 1945). We denote the number of active pins on the wheels as N_1 to N_6 and list the conditions:

1. N_1 is not divisible by 5
2. $N_2 \neq 21$
3. $N_3 \neq 0$ and $N_3 \neq 23$
4. $N_4 \neq 1 \mod 2$ and $N_4 \neq 1 \mod 3$
5. N_5 is neither divisible by 2 nor by 3

In (Agency, 1947), an example is given demonstrating that by violating only one of the five conditions, it is possible to obtain a motion period with only 70 stepping stages. Generally, if the conditions are not systematically followed, the vast

majority of (randomly-selected) settings would result in periods shorter than the maximum period. It can be seen that condition 1 leaves only 0.8 of the possible wheel settings, condition 4 one-third of those, and condition 5 leaves one-third of the latter, so that they may generate a complete period. With just those three conditions, we are left with 0.8/9, which is less than one-tenth. Therefore, more than 90% of randomly selected settings would be sub-optimal, and even if the motion period is longer than one million, on a day with heavy traffic, with tens of thousands of symbols intercepted, overlaps (partial depths) are likely to occur.

The Bletchley Park report, and a report written by Walter Fried, the U.S. liaison officer in Bletchley Park, states that no generic solution could be devised to read SG-41 traffic, for messages not in-depth. Furthermore, even the availability of a crib, or plaintext recovered from depths, did not allow for the reconstruction of the key settings (Batey et al., 1945; Fried, 1944).

4 A Novel Known-Plaintext Attack

Because of the complex stepping mechanism, there are no periodic patterns that would allow statistical attacks to be effective, such as the ciphertext-only and known-plaintext attacks that were developed against Hagelin systems with regular stepping (Lasry et al., 2016; Lasry et al., 2018). Furthermore, the circular dependencies of the wheels with regards to their stepping (as illustrated in Figure 4) make the problem even more challenging. Basically, to know how the wheels will step, one needs to know all the pin settings. But to recover the pin settings, one needs to know how the wheels step.

To break that circular dependency, one approach is to exhaustively test some of the elements of the circular logic chain and to validate the elements under test and/or reconstruct additional elements further in the logical chain. There is a trade-off between the number of options to test and the number of elements under test. On the one hand, the richer the information in the elements under test, the easier it is to rule out wrong options in an efficient manner. On the other hand, more elements under test means that more options need to be tested.

Finding the right balance between the number of options to test and the amount of information that allows for a definitive evaluation requires extensive trial-and-error with various testing scenarios. In this section, we present a recursive, backtracking algorithm to validate candidate settings of wheels 1 and 6, based on a sequence of effective kick K_e (obtained from the ciphertext and known-plaintext). The technique is illustrated in Figure 5. It not only tests the settings of wheels 1 and 6 but also reconstructs the settings of wheel 2. The algorithm starts with unknown states for all the pins of wheels 2. It recursively processes the sequence of K_e, testing all possible T5 options at each encryption cycle, advancing wheels 1, 2, and 6 accordingly. It then validates candidate C1 and C6 against K_e and Table 1 (C1 can always be determined unambiguously from C6 and K_e), and tries to deduce C2 from K_e. If C2 can be deduced unambiguously, the pin at the current C2 position is updated accordingly. If C2 at that position has already been updated (this is possible if the wheel has already rotated once or more), then the algorithm validates that there is no conflict. If there is a conflict, the algorithm discards the option for T5, and if all T5 options have been discarded, it backtracks. If there is no conflict (neither with C1 nor with C2), the algorithm recursively processes the next encryption cycle and its associated K_e. If this is the last encryption cycle for which there is known-plaintext, and there are no more K_e elements to process, the (tested) settings of wheels 1 and 6 and the reconstructed settings of wheel 2, constitute a solution candidate.

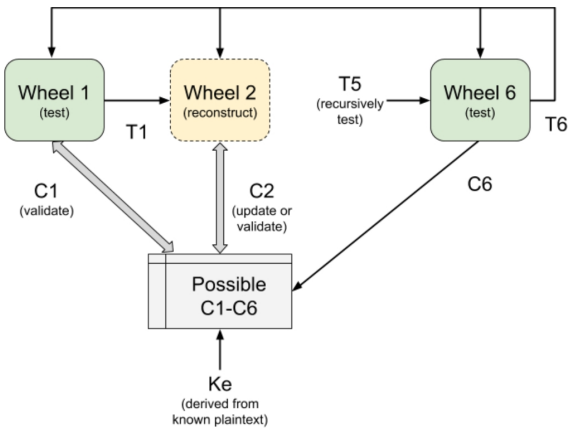

Figure 5: Known-Plaintext Attack – Recovering Wheel 2 Settings

We present here the algorithm to recover the settings of wheel 2 from the settings of wheels 1 and 6 and from known-plaintext. Its complexity reflects the complexity of the stepping logic.

Recursive procedure:

1. Repeat (2) to (5) for all possible pin settings of wheels 6 and wheels 1. There are $2^{25+24} = 2^{49}$ such settings.

2. Initially mark the state of all the pins of wheel 2 as *unknown*. We will mark them as *active* or *inactive* as we gather unambiguous evidence in the procedure described here.

3. Assume that the starting position of all wheels is pin 1 (another position may be assumed, and the algorithm would produce equivalent, shifted, pin settings).

4. Start by processing the first encryption cycle (the first ciphertext and known-plaintext symbols).

5. Determine T6 (from the wheel 6 settings under test, at the pin currently driving T6):

   (a) If T6 is active, the optional pre-encryption stepping stage is applied:
      i. Advance wheels 1, 2, and 6, and advance again wheel 2 if T1 was active.
      ii. For each possible state of pre-encryption T5 (active or inactive):
         A. Advance wheel 6 if T5 is active.
         B. Validate/update C1 and C2 (see below). If a conflict is detected, discard this option for T5, or backtrack if both T5 options result in a conflict.
         C. Advance wheels 1, 2, and 6, and advance again wheel 2 if T1 was active.
         D. For each possible state of (post-encryption) T5 - active or inactive, advance wheel 6 if T5 is active, and recursively perform (5) for the next encryption cycle (the next ciphertext and plaintext symbols). If this is the last encryption cycle (for which there is known-plaintext), store the settings of wheel 2 as a candidate solution.

   (b) If T6 is inactive, only the post-encryption stepping stage is relevant:
      i. Validate/update C1 and C2 (see below). If a conflict is detected, discard this option for T5, or backtrack if both T5 options result in a conflict.
      ii. Advance wheels 1, 2, and 6, and advance again wheel 2 if T1 was active.
      iii. For each possible state of (post-encryption) T5 - active or inactive, advance wheel 6 if T5 is active, and recursively perform (5) for the next encryption cycle (the next ciphertext and plaintext symbols). If this is the last encryption cycle (for which there is known-plaintext), store the settings of wheel 2 as a candidate solution.

Procedure to validate/update C1 and C2:

1. Compute K_e, the effective kick for the current ciphertext and known-plaintext symbol.

2. Determine the expected state of C1 from K_e and the current C6 (C1 can be determined unambiguously - see Table 1). If the expected state of C1 is different from the state of C1 at the current position (based on the wheel 1 settings being tested), the procedure fails.

3. Update or validate C2, as follows:

   (a) If K_e is between 0 and 9, or 16 and 25, it is possible to determine the state of C2 unambiguously from K_e and the current C6 (see Table 1).
      i. If the state of C2 at the current position was previously marked as active or inactive, verify that it does not conflict with the C2 derived from K_e and C6. If there is a conflict, the procedure fails.
      ii. If the state of C2 at the current position was previously marked as unknown, update it with C2 derived from K_e.

   (b) If K_e is between 10 and 15, it is not possible to determine C2 unambiguously from K_e and the current C6. No update or validation for C2 can be done in this encryption cycle.

When processing the initial known-plaintext symbols, the state of C2 at the current position can only be updated and not validated, as there is no prior knowledge. As wheel 2 completes a full rotation, previously updated C2 states can be compared with C2 states newly derived from K_e, checking for contradictions and pruning wrong T5 assumptions, or wrong options under test (settings of wheels 1 and 6). If SG-41 were designed so that wheels advance only once or twice (versus up to four times) per encryption cycle, this attack would have been less effective.

A similar technique is employed to recover the pin settings of wheel 3 from the pin settings of wheels 6, 1, and 2. Similarly, the pin settings of wheels 4 and 5 can be recovered. The candidate pin settings that survive all the algorithm stages are finally verified by decrypting the ciphertext and ensuring that the resulting decryption indeed matches the known-plaintext.

To rule out all wrong settings of wheel 2, a crib of about 150 symbols is required. However, a crib of 80 symbols is enough to rule out most of the wrong wheel 2 settings, while the additional phases (recovering wheel 3, wheel 4, and wheel 5) can discard the remaining wrong ones.

The first phase of the algorithm needs to test 2^{49} options, for all possible settings of wheels 6 and 1. Subsequent phases - for recovering the pins of rotors 3 to 5 - require only 2^{23} to 2^{25} runs. Based on preliminary benchmarks, it is estimated that a few thousands of PCs would complete the process in a month. Further research is needed to evaluate whether additional optimizations or a GPU implementation could further speed up the process.

5 Possible Side-Channel Acoustic Attack on SG-41

The SG-41 is a purely mechanical machine. As wheels step up to four times, it is also a noisy machine, as can be heard in a video of a machine recently restored (Historica, 2019). The sound emitted by the machine is likely to leak extensive information about its internal functioning, and wheel stepping in particular. In this attack, we assume that it is possible to determine at each encryption cycle, based on the sound generated by the SG-41, whether there was one or two stepping stages (that is, whether the optional pre-encryption stepping stage took place). In other words, it is possible to extract a sequence of T6 states from an acoustic recording. This assumption has not been checked yet against a real machine, but it is highly plausible.

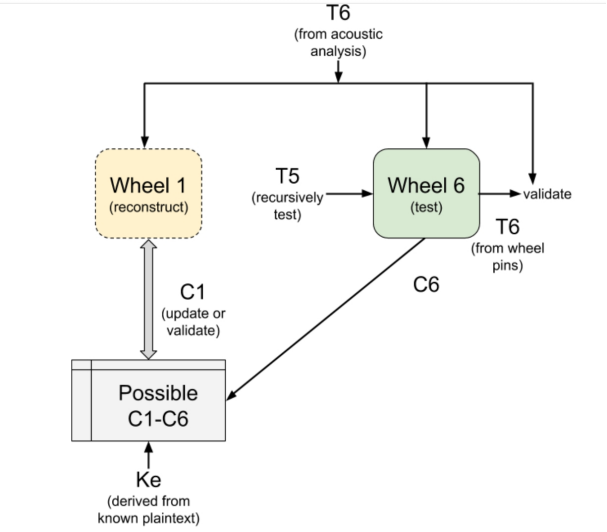

Figure 6: Acoustic Attack – Recovering Wheel 1 Settings

The process for recovering the wheel settings is similar to the process described in Section 4, and only its outline is presented here. The algorithm is applied to all possible settings of wheel 6 (there are 2^{24} such settings). The recursive backtracking algorithm is illustrated in Figure 6. It recursively tests all T5 options at each encryption cycle and it recovers the states of wheel 1 pins, based on C1 that can be derived from K_e and C6 (after wheel 1 completes a full rotation, contradictions can also be detected). It also checks whether T6 derived from wheel 6 pins matches the T6 pattern predicted via acoustic analysis. If there is a conflict (in either C1 or T6), it backtracks. If there is no conflict, the next encryption cycle (the next known-plaintext and ciphertext symbols) is recursively processed.

After candidate settings of wheel 1 have been recovered, the settings of wheels 2 are similarly recovered, and so forth for the remaining wheels. With about 100 known-plaintext symbols (and the relevant T6 sequence), only a handful of candidate solutions survive the last stage of the algorithms, and the wrong ones can be eliminated with a simple decryption test. The algorithm takes a few minutes to test all possible wheel 6 settings.

6 Conclusion

The functional description of the SG-41 in this article is based on historical British and U.S. reports(Batey et al., 1945; Mowry, 1989; Mowry, 2003; Mowry, 1983), and on information that has been made available recently, following the work of curators and collectors who own a SG-41 (Museum, 2020b; Historica, 2019; Dahlke, 2018; I-72, 1945). The acoustic attack described in Section 5 might need to be refined, based on further analysis of the precise information that may leak acoustically, but based on our work, we can provide an revised assessment of the security of the SG-41.

The attack described in Section 4, when nothing is known except for a segment of plaintext, requires 2^{49} runs of the core algorithm. Taking into account the complexity of the core algorithm, the author estimates that the security of the SG-41 is comparable to a modern cipher with 60-bit key (DES as a 56-bit key). The fact that a significant amount of processing power is required for its cryptanalysis with modern techniques is a testimony to the high level of security of the device, compared to other WWII German and Allied cipher machines. It is much more secure than Enigma, and probably provides the same level of security as SIGABA and T52e, the most sophisticated cipher machines of the time (Lasry, 2019). An historical report by the Army Security Agency even suggested designing a new device based on the same principles as the SG-41, to be used by the U.S. (WDGAS-14, 1946; Mowry, 1983).

Some of the features of the SG-41 such as irregular stepping with circular dependencies, and the complementary feature, are nowhere to be seen in other devices, until the 1950s, with some advanced models of the Hagelin CX-52 (Museum, 2020b; Friedman, 1955).

Acknowledgments

This work has been supported by the Swedish Research Council, Grant 2018-06074, DECRYPT – Decryption of historical manuscripts. In addition, the author would like to thank Ralph Erskine and Frode Weierud for providing access to key historical documents and for reviewing an early version of this article, and Klaus Kopacz for clarifying some of the details of the SG-41 stepping mechanism.

References

Army Security Agency. 1947. *Observations on C-41 Cycles. Coll. 354: STINFO, Box 278, Folder16380*. NARA, College Park, MD.

Keith Batey, Mavis Batey, Margaret Rock, and Peter Twinn. 1945. *Secret Service SIGINT Volume II – Cryptographic Systems and Their Solutions – Machine Cyphers*. G.C. & C.S., TNA, Kew, HW 43/7.

Carola Dahlke. 2018. What We Know About Cipher Device "Schlüsselgerät SG-41" so Far. In *Proceedings of the 1st International Conference on Historical Cryptology HistoCrypt 2018*, number 149, pages 109–111. Linköping University Electronic Press.

Walter J. Fried. 1944. *ISK Report, F-119, IR 4065*. War Department.

William F. Friedman. 1955. *Report of Visit to CRYPTO A.G. (Hagelin)*. National Security Agency.

Hermann Historica. 2019. *YouTube Video: Schlüsselgerät 41 - Cipher Machine 41 (SG-41)*.

TICOM I-72. 1945. *First part of the report by Wm. Buggisch on S.G.41 (still classified)*.

George Lasry, Nils Kopal, and Arno Wacker. 2016. Automated Known-Plaintext Cryptanalysis of Short Hagelin M-209 Messages. *Cryptologia*, 40(1):49–69.

George Lasry, Nils Kopal, and Arno Wacker. 2018. Ciphertext-Only Cryptanalysis of Short Hagelin M-209 Ciphertexts. *Cryptologia*, 42(6):485–513.

George Lasry. 2019. A Practical Meet-in-the-Middle Attack on SIGABA. In *2nd International Conference on Historical Cryptology HistoCrypt 2019*, page 41.

David P. Mowry. 1983. Regierungs-Oberinspektor Fritz Menzer: Cryptographic Inventor Extraordinaire. *Cryptologic Quarterly*, Volume 2, Nos. 3-4.

David P. Mowry. 1989. *The Cryptology of the German Intelligence Services*. United States.

David P. Mowry. 2003. *German Cipher Machines of World War II: Description Based on Print Version Record*. National Security Agency, Center for Cryptologic History.

Crypto Museum. 2020a. *Hagelin C-35*. www.cryptomuseum.com/crypto/hagelin/c35/index.htm.

Crypto Museum. 2020b. *Schlüsselgerät 41*. www.cryptomuseum.com/crypto/sg41/index.htm.

Klaus Schmeh. 2004. *Hitlers letzte Maschinen*. Telepolis. Heise.

WDGAS-14. 1946. *European Axis Signal Intelligence in World War II – Volume 2: Notes on German High Level Cryptography and Cryptanalysis*. Army Security Agency.